



Buscando nuevas maneras de enseñar a programar...

Manual de Usuario
PLUGIN ESTACIÓN TRENES

Tabla de Contenidos

1. REQUERIMIENTOS

1.1 Hardware

Los mismos de la plataforma Eclipse.

1.2 Software

- ✓ Sistema operativo Windows
- ✓ Eclipse: 3.1.0 o superior
- ✓ JDK 1.5 o superior

2. INSTALACIÓN

1) Baje el plugin del sitio: http://cupi2.uniandes.edu.co/cursos/apo1/nivel3.htm
2) Cierre la plataforma Eclipse (en caso de que la tenga abierta).
3) Descomprima el archivo "co.edu.uniandes.cupi2.E_Trenes_Arreglos_1.0.zip" que bajo en el punto anterior en la carpeta de plugins de Eclipse. El path de esta debe ser ../Eclipse/plugin.
4) Abra la plataforma Eclipse. Eliga En el menú `Windows>Open Perspectiva>Other` la perspectiva "Estacion Trenes". Si le aparece una ventana similar a la siguiente, la perspectiva ha sido instalada con éxito.

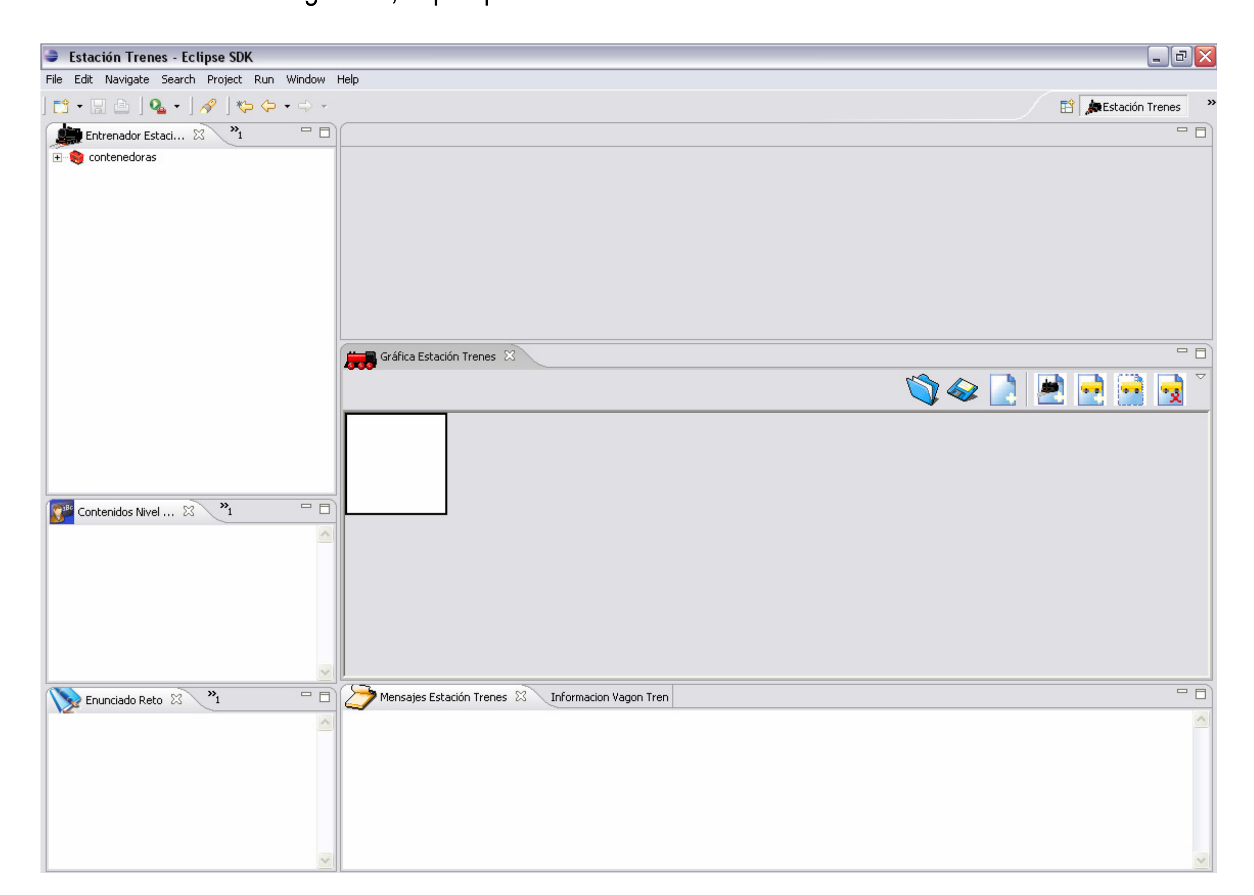


Figura 1. Pantalla inicial de la perspectiva



3. FAMILIARIZACIÓN CON LAS VISTAS DEL PLUGIN

El plugin consta de 8 vistas las cuales pueden ser apreciadas en la figura 2a y 2b. A continuación se realiza una breve explicación de cada una de ellas.

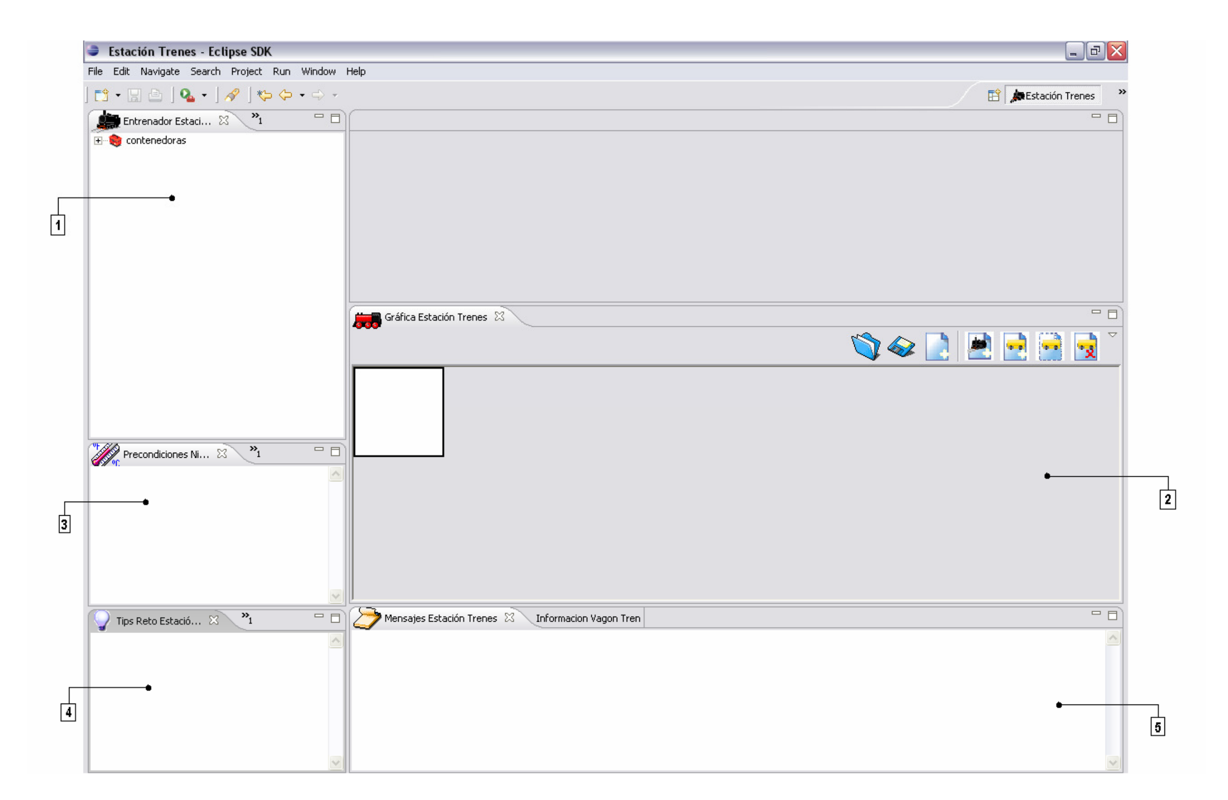


Figura 2a. Partes de la perspectiva

1. Explorador del entrenador
2. Vista de la gráfica de los trenes
3. Vista de las precondiciones del nivel
4. Vista de los tips de los retos
5. Vista de los mensajes de los trenes



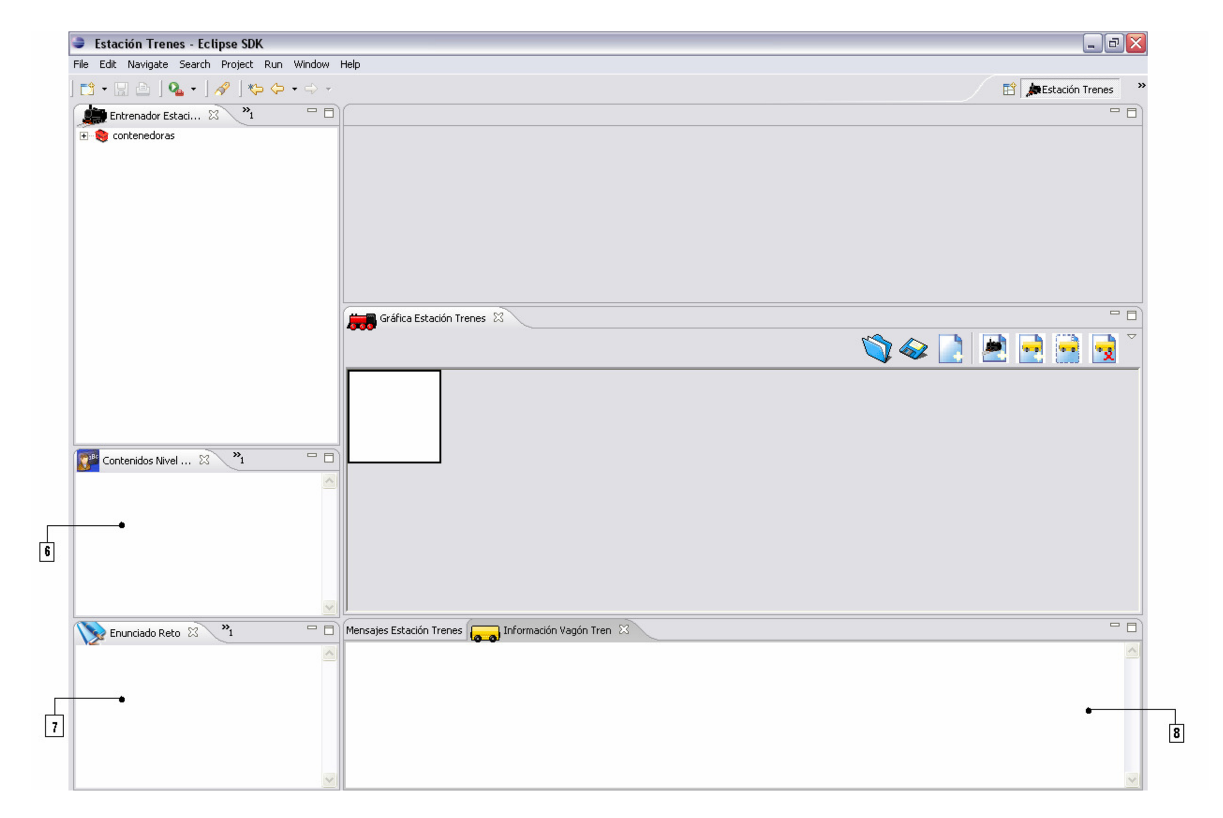


Figura 2b. Partes de la perspectiva

6. Vista de los contenidos del nivel
7. Vista de los enunciados de los retos
8. Vista de la información de los vagones

3.1 Explorador del entrenador

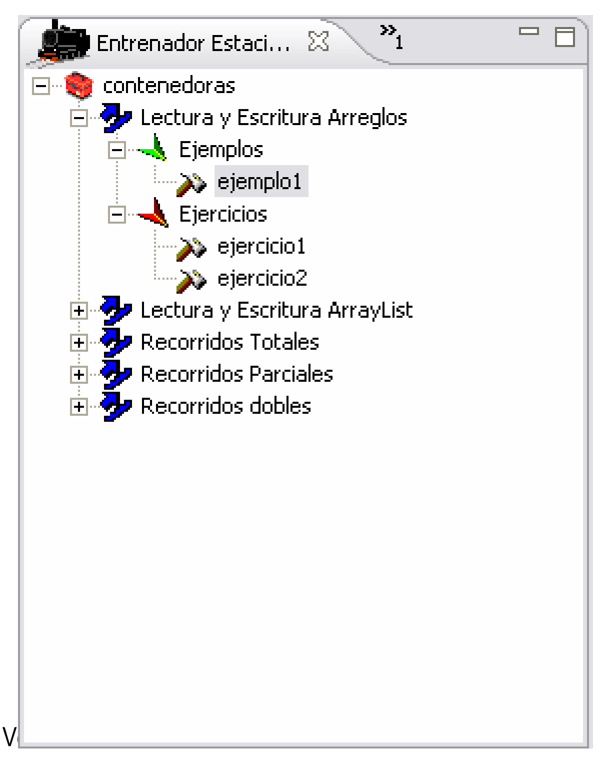


Figura 3. Explorador de Estación Trenes

Por medio de esta vista se tiene la posibilidad de explorar los diferentes niveles y retos que se encuentran definidos en el entrenador. Cuando se hace click sobre un nivel, se muestran los contenidos y precondiciones asociadas con estos en las vistas respectivas. Cuando se elige un reto de entrenamiento, se despliegan el enunciado y los tips asociados con este en las vistas correspondientes; también se muestra el tren (en la vista de la gráfica de los trenes) relacionado con dicho reto. Además, por medio de esta vista se pueden crear y ejecutar soluciones para los retos predefinidos (ver secciones 4.1 y 4.2). En la figura 3 se muestra la vista explorador del entrenador.



3.2 Vista de la gráfica de los trenes

En esta vista se despliega la gráfica de los trenes. Además, permite la manipulación del tren actual ya que brinda la posibilidad de adicionar, eliminar y modificar vagones.

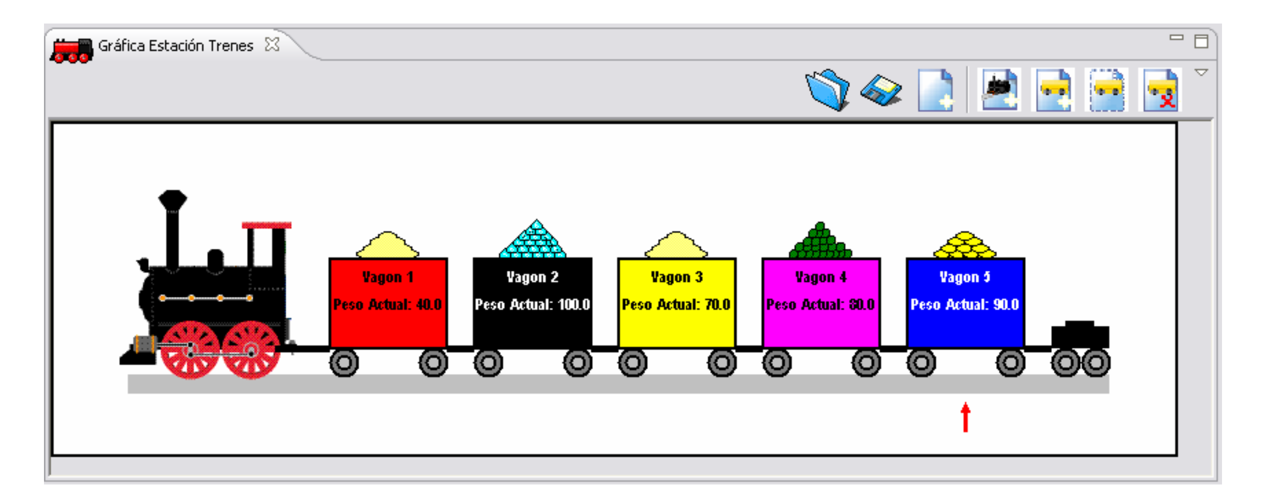


Figura 4. Vista de la gráfica de los trenes

3.2.1 Principales características y restricciones de la estación de trenes

La estación está compuesta principalmente por uno o más trenes. Sin embargo en la versión 1.0 del plugin sólo es posible visualizar el último tren que haya sido creado en la vista. Cuando la estación está vacía (es decir no tiene tren alguno) la grafica despliega un cuadrado blanco con contorno negro (ver figura 1).

Adicionalmente la estación puede almacenar ArrayList de enteros. Éstos son desplegados en la gráfica como recuadros con los números correspondiente a cada posición (ver figura 5).

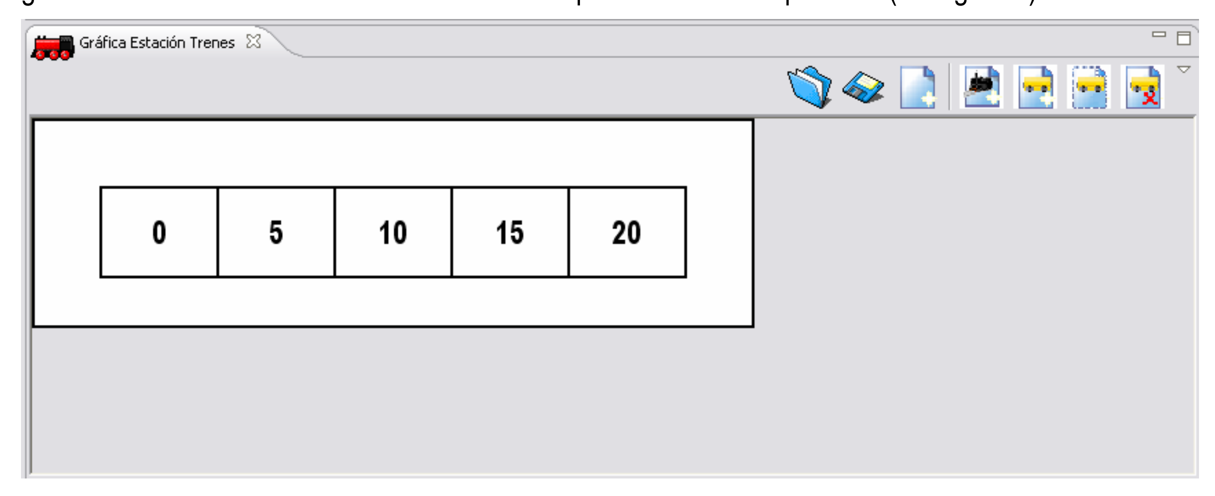


Figura 5. Vista de la gráfica de los trenes desplegando un objeto de tipo ArrayList.



3.2.2 Botones de la vista

La vista cuenta con 7 botones, cuya funcionalidad se explica a continuación.



Cargar Estación

Permite el despliegue en la gráfica de una estación descrita en un archivo. Como ya se ha mencionado, sólo es desplegado el último tren de la estación. Cuando se hace click sobre este botón se despliega un dialogo de archivo, como el que se muestra en la figura 6, el cual sólo permite abrir archivo cuya extensión es ".station". Dicho dialogo de archivo se abre por defecto en la ruta `c:\temp\estaciones`; en el caso de que dicha ruta no exista se abre en la carpeta "Mis documentos".

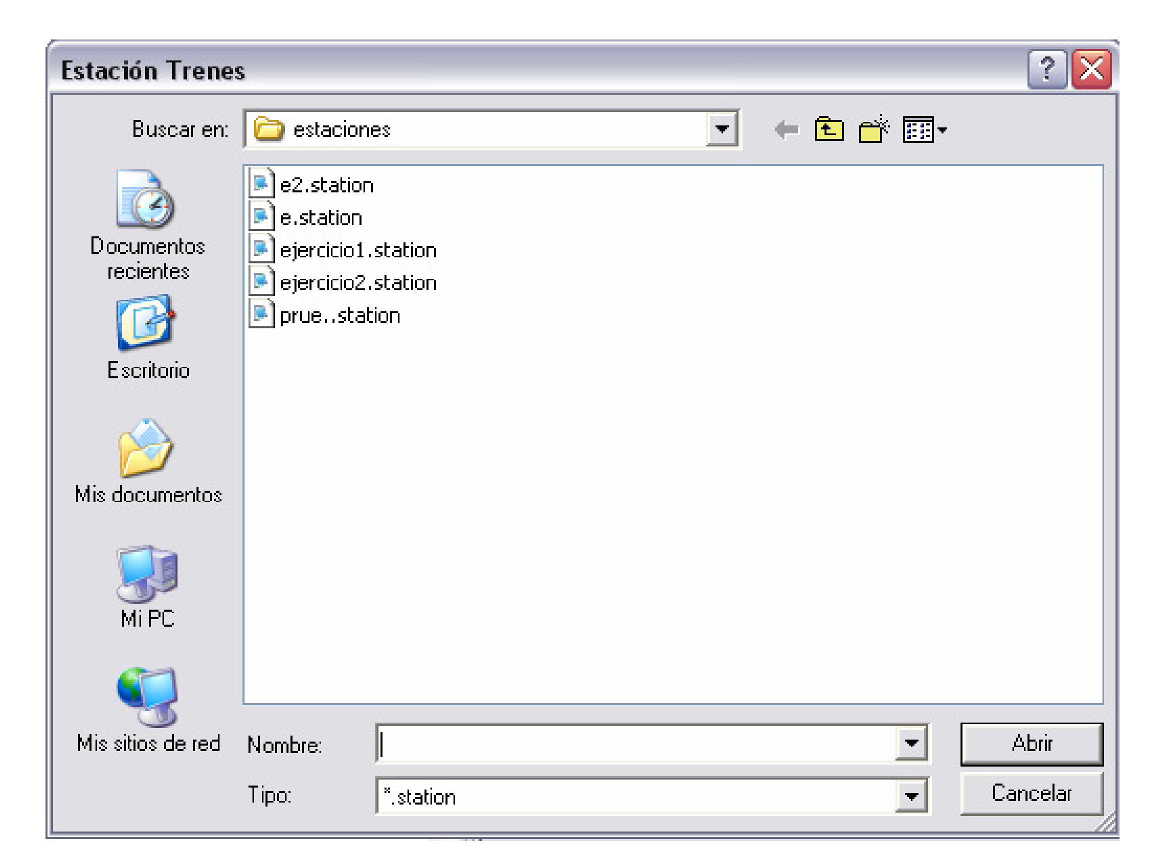


Figura 6. Dialogo para cargar un archivo ".station"



Guardar Estación

Permite guardar el estado de la estación actual en un archivo con extensión ".station". Cuando se hace click sobre este botón se despliega un dialogo de salvar como el que se muestra en la figura 7.

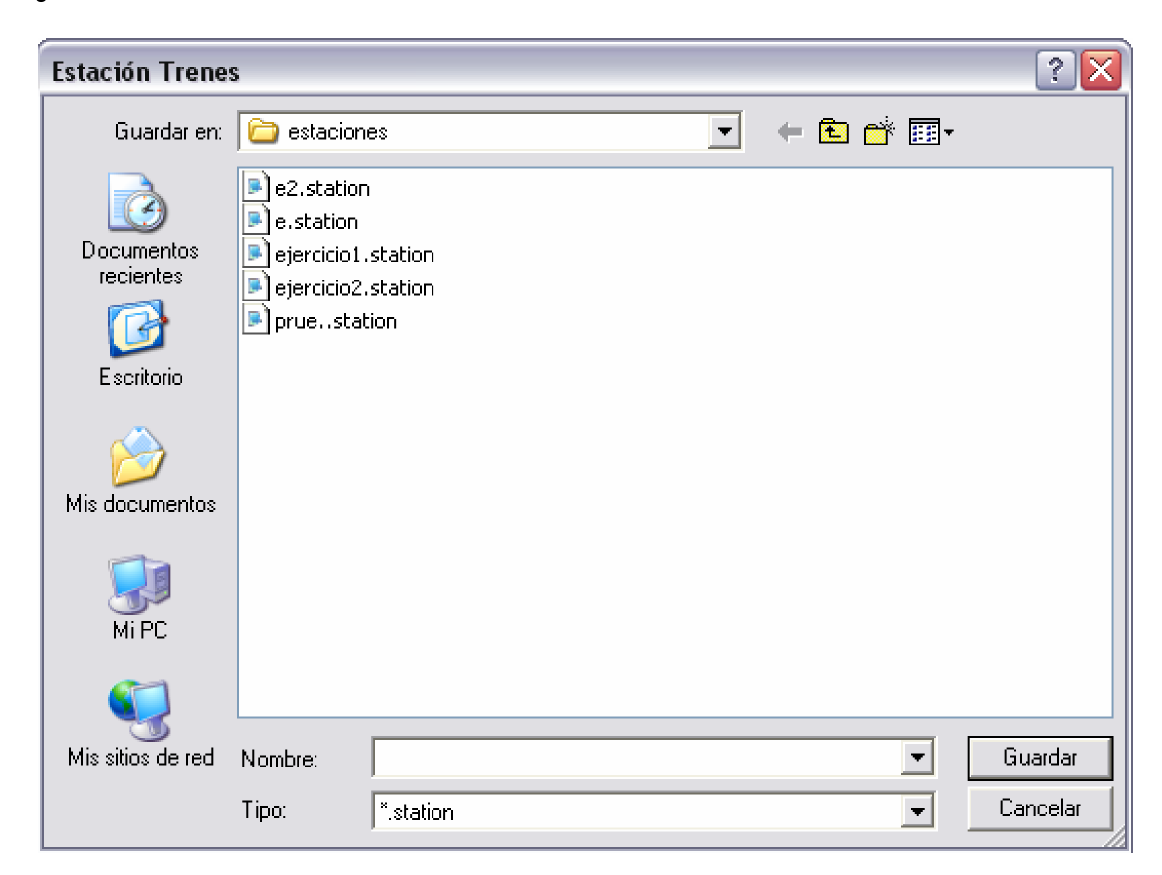


Figura 7. Dialogo para guardar el estado de una estación en un archivo ".station"



Borrar Estación

Al hacer click sobre este botón se borran todos los trenes que existen en la estación y por tanto en la gráfica se muestra un recuadro blanco vacío.



Crear Tren

Permite la creación de un tren. Al hacer click sobre este botón se despliega una ventana como la que se muestra en la figura 8. Para crear un tren se debe seleccionar el número de vagones que éste va a tener. El número máximo de vagones permitidos es 100. Todos los vagones del tren



son creados con los valores por defecto (color blanco, contenido vacío, sin nombre, altura 0, ancho 0, peso actual 0 y peso máximo 0).

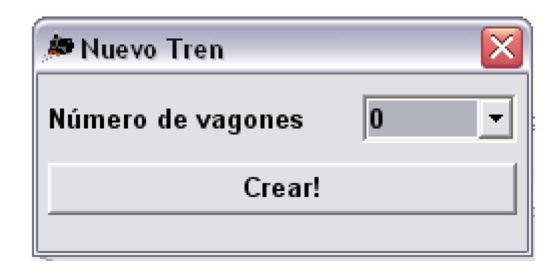


Figura 8. Ventana para la creación de un tren



Adicionar Vagon



Permite la adición de un vagón al tren que está siendo desplegado en la gráfica. Al hacer click sobre este botón aparece una ventana similar a la que se muestra en la figura 9a. Para adicionar un vagón es necesario especificar su nombre, contenido, color, alto, ancho, peso actual y peso máximo. En caso de que no existe un tren en la estación y se intente adicionar un vagón, se despliega un mensaje similar al de la figura 9b.

Figura 9a. Ventana para la adición de un vagón

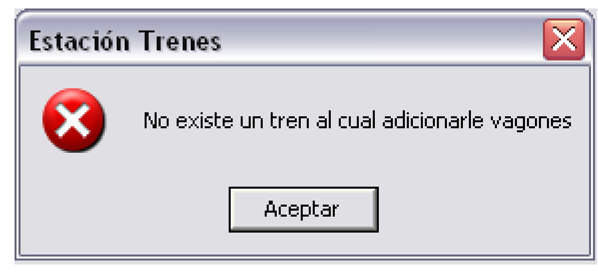
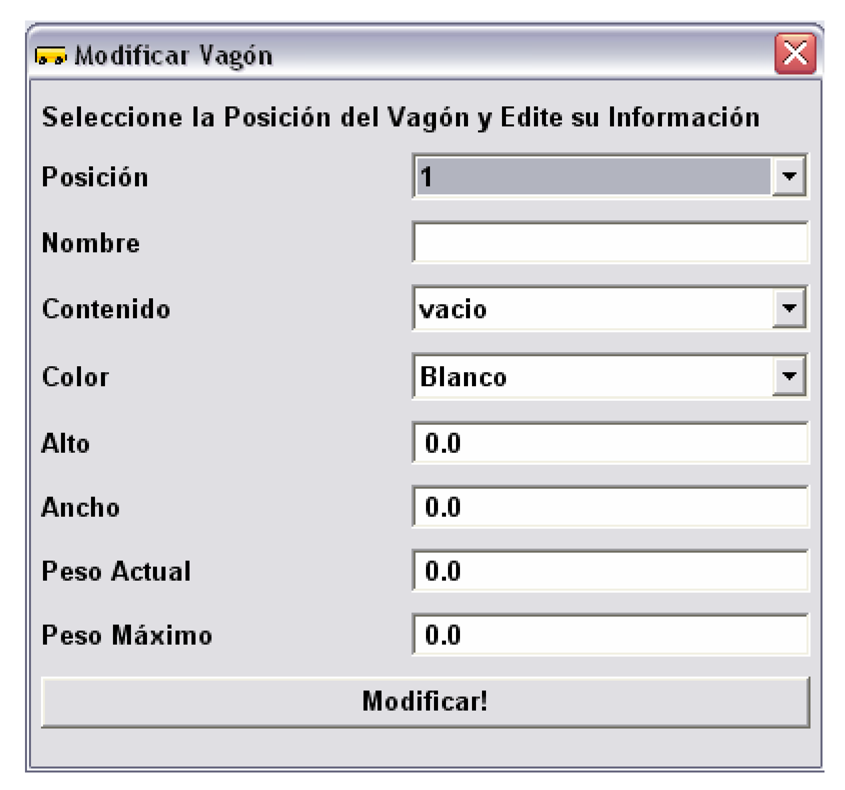


Figura 9b. Mensaje de error al intentar adicionar un vagón cuando no existen trenes





Permite modificar la información relacionada con un vagón existente en el tren que se está desplegando. Al hacer click sobre este ítem se despliega la ventana mostrada en la figura 10a. Para modificar un vagón es necesario seleccionar la posición del mismo.

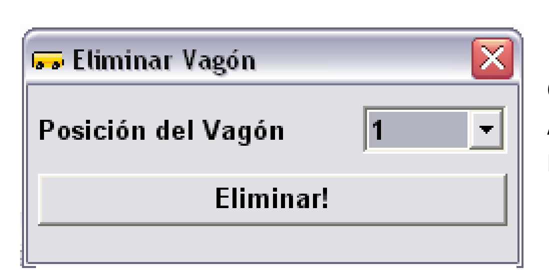
En caso de que se trate de modificar un vagón sin que haya trenes en la estación se despliega un mensaje como el que aparece en la figura 10b. Si el tren no tiene vagones, no es posible seleccionar la posición de vagón alguno

Figura 10a. Ventana para modificar un vagón



Figura 10b. Mensaje de error al intentar modificar un vagón cuando no existen trenes

Eliminar Vagón



Permite eliminar un vagón existente en el tren que se está desplegando actualmente, indicando su posición. Al seleccionar este ítem aparece la ventana que se muestra en la figura 11a.

Figura 11a. Ventana para eliminar un vagón



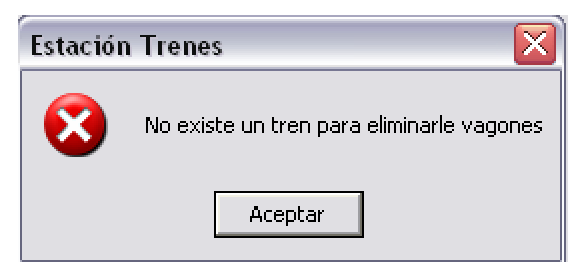


Figura 11b. Mensaje de error al intentar eliminar un vagón cuando no existen trenes

En caso de que se trate de eliminar un vagón sin que se haya un tren, se despliega el mensaje mostrado en la figura 11b. Si el tren no tiene vagones, no es posible seleccionar la posición de vagón alguno.

3.2.3 Menús de la vista

La vista de gráfica de los trenes se compone de dos menús: “Estación” y “Trenes”.

<u>Menú Estación</u>

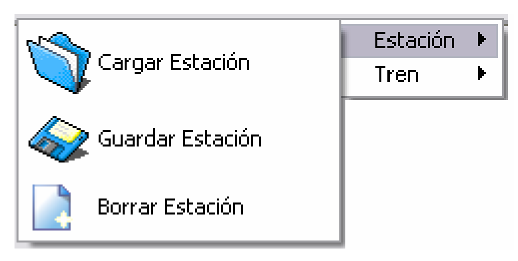


Figura 12. Menú Estación

En éste se encuentra los Items de “Cargar Estación”, “Guardar Estación”, y “Borrar Estación“La función que cumplen estos ítems es idéntica a los botones correspondientes descritos en la sección 3.2.2.

<u>Menú Trenes</u>

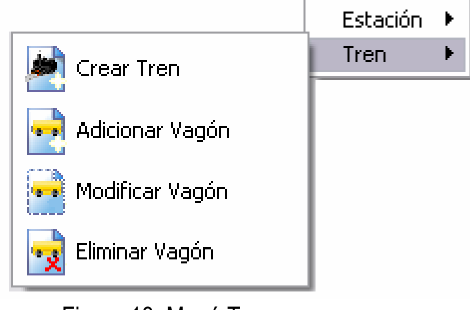


Figura 13. Menú Tren

En éste menú se encuentran los ítems de “Crear Tren”, “Adicionar Vagón”, “Modificar Vagón” y “Eliminar Vagón”. La función que cumplen estos ítems es idéntica a los botones correspondientes descritos en la sección 3.2.2.



3.3 Vista de las precondiciones del nivel

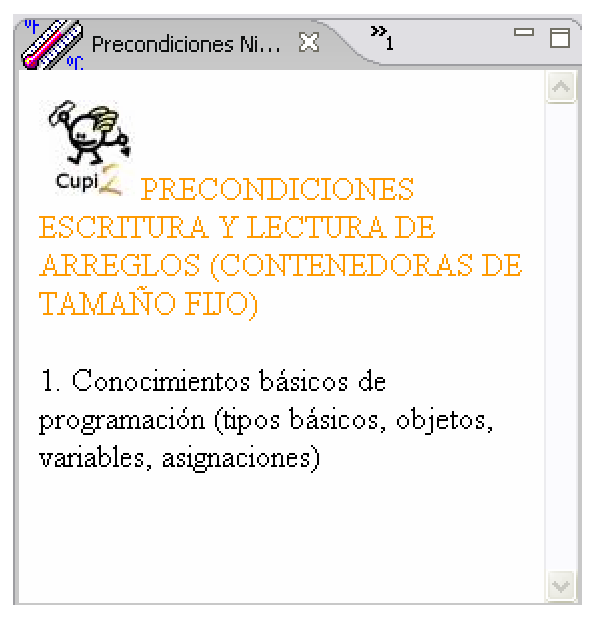


Figura 14. Vista de precondiciones.

Esta vista permite la visualización de las precondiciones asociadas con un nivel. Para visualizar las precondiciones en esta vista es necesario hacer click en el explorador sobre el nivel de interés. En la figura 14 se puede apreciar la vista de precondiciones.

3.4 Vista de los tips de los retos

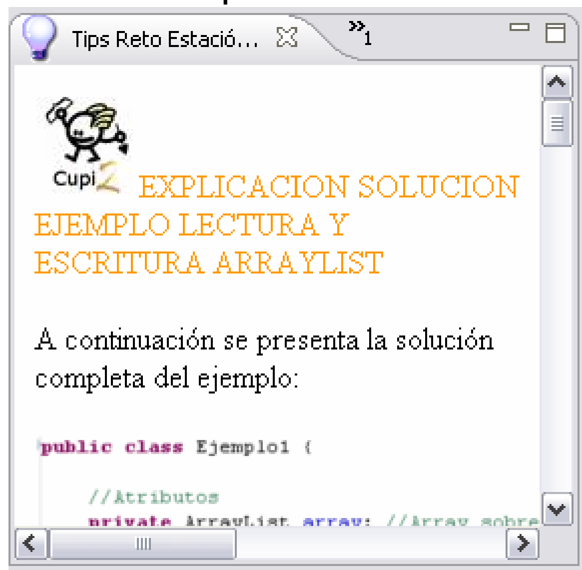


Figura 15. Vista de tips.

Esta vista permite la visualización de los tips asociados con un reto de entrenamiento. En el caso de los ejemplos, se trata de una explicación de la solución y en el caso de los ejercicios de ayudas para el desarrollo de los mismos. Para visualizar los tips de un reto en esta vista es necesario hacer click en el explorador sobre el ejemplo o ejercicio de interés. En la figura 15 se puede apreciar la vista de tips.



3.5 Vista de información de los vagones

Al hacer click sobre uno de los vagones del tren que se está desplegando, en esta vista se despliega la información relacionada con dicho vagón. En la figura 16 se puede apreciar la vista de información de los vagones.



Figura 16. Vista de información de los vagones

3.6 Vista de los contenidos del nivel

Esta vista permite la visualización de los contenidos asociados con un nivel. Para visualizar los contenidos en esta vista es necesario hacer click en el explorador sobre el nivel de interés. En la figura 17 se puede apreciar la vista de contenidos.

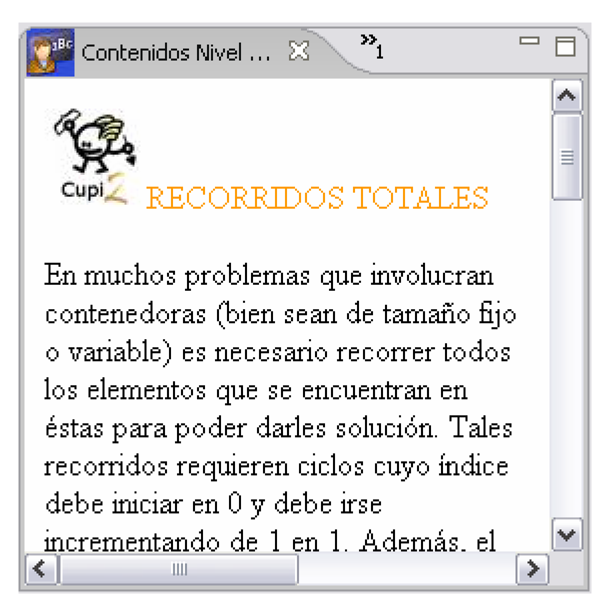


Figura 17. Vista de contenidos.



3.7 Vista de los enunciados de los retos

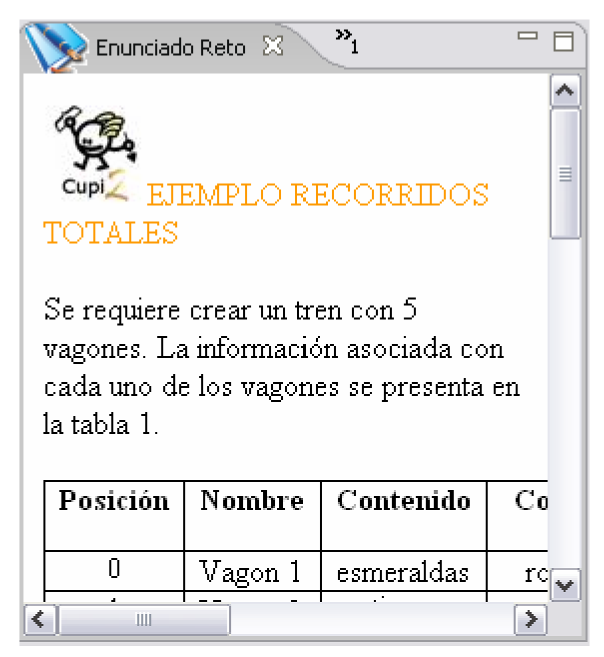


Figura 18. Vista de enunciados.

Esta vista permite la visualización del enunciado asociado con un reto de entrenamiento. Para visualizar el enunciado de un reto en esta vista es necesario hacer click en el explorador sobre el ejemplo o ejercircio de interés. En la figura 18 se puede apreciar la vista de enunciados.

3.8 Vista de los mensajes de los trenes

En esta vista se despliegan los mensajes asociados con las acciones que se realizan sobre los trenes.



Figura 20. Vista de mensajes

3.9 Vistas adicionales

La perspectiva Estación Trenes utiliza dos vistas de Java. La primera de ellas, es el package explorer (ver figura 21a) que permite la exploración de los proyectos que se crean como proyectos Java. La otra vista es el editor, también de Java, que permite que el estudiante desarrolle la solución de los retos planteados (ver figura 21b).

Figura 21. Vistas adicionales de la perspectiva: a) Package Explorer. b) Editor

4. TRABAJANDO CON LOS RETOS DE ENTRENAMIENTO

A parte de visualizar los enunciados y tips asociados con un reto de entrenamiento es posible crear, desarrollar y ejecutar las soluciones asociadas con estos.

4.1 Creando una nueva solución

Para la creación de una solución asociada con un ejemplo o ejercicio siga los pasos que se enuncian a continuación:

1. Sitúese en la vista explorador del entrador sobre el reto al que desea crearle una solución.
2. Haga click derecho sobre éste. Le debe aparecer un menú como el que se muestra en la figura 22a.
3. Elija la opción “Crear Solución”, haciendo click sobre ésta. Le debe aparecer una ventana como la que se muestra en la figura 22b.
4. Digite el nombre del proyecto y del archivo.
5. Haga click en “Crear!”.

Si el proyecto que digita no existe, éste es creado. En el caso de que exista, el archivo es creado dentro de éste. Cuando ingresa un proyecto y archivo que ya existen pero que no se encuentran asociados con el reto sobre el que está creando la solución, dicho archivo es asociado con el reto (sin borrarlo ni volverlo a crear).

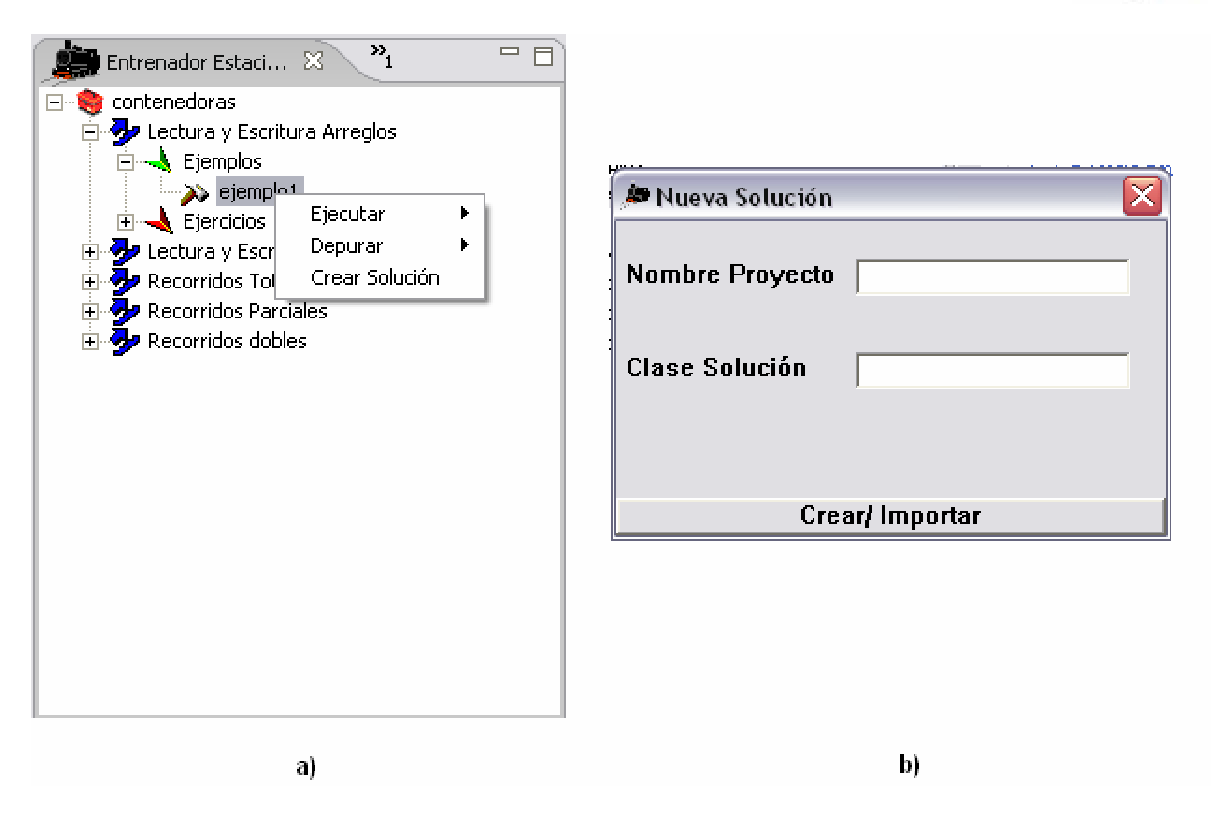


Figura 22. Componentes involucrados en la creación de una solución: a) Explorador del entrenador. b) Ventana para la creación de una solución

4.2 Ejecutando la solución de un reto

Para la ejecución de una solución asociada con un reto siga las siguientes instrucciones:

1. Sitúese en la vista explorador del entrador sobre el reto al que desea ejecutarle una solución.
2. Haga click derecho sobre éste. Le debe aparecer un menú como el que se muestra en la figura 23.
3. Al situarse sobre la opción "Ejecutar" le deben aparecer todas las soluciones asociadas con el reto (ver figura 23).
4. Haga click sobre la solución que desee ejecutar. La ejecución deberá iniciar inmediatamente.

A los ejemplos se les crea por defecto un archivo que contiene su solución en un proyecto bajo el nombre "uniandes.cupi2.entrenadores.trenarreglos.retos", de manera que éstos siempre deberían tener una solución asociada. Usted puede modificar tales soluciones. Con respecto a los ejercicios, es necesario crearles la solución (ver sección 4.1).

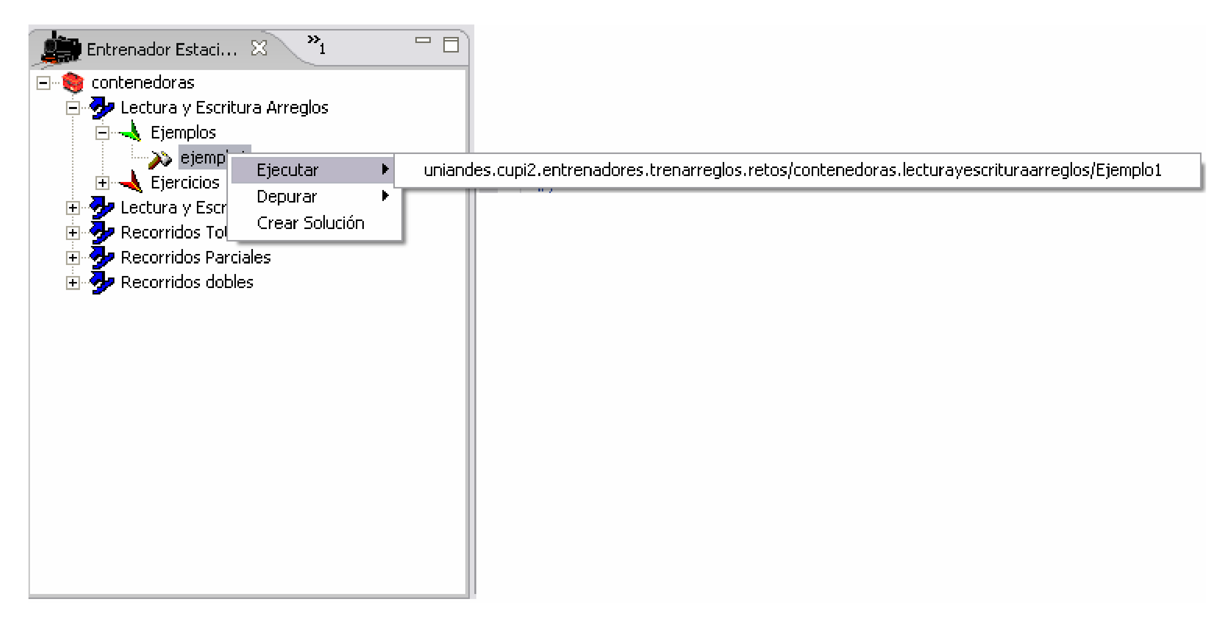


Figura 23. Forma de ejecutar una solución de un reto predefinido

4.3 Haciendo Debug a la solución de un reto

Para la realización de debug sobre una solución siga los siguientes pasos:

1. Sitúese sobre el archivo solución sobre el que desea hacer debug.
2. Ponga un break point en la línea desde la que desea hacer el debug.
3. Sitúese en la vista explorador del entrador sobre el reto al que desea hacerle debug a una de sus soluciones.
4. Haga click derecho sobre éste. Le debe aparecer un menú como el que se muestra en la figura 24a.
3. Al situarse sobre la opción “Debug” le deben aparecer todas las soluciones asociadas con el reto (ver figura 24a). Si el reto no tiene soluciones asociados, debe crear y desarrollarle una.
4. Haga click sobre la solución sobre la que desee hacer debug. Le debe aparecer un diálogo (ver figura 24b) preguntándole si desea cambiar de perspectiva, elija “No”.
5. En la perspectiva del entrenador se debe abrir una vista llamada “Debug”(ver figura 25)
6. Para avanzar en la solución paso a paso utilice la tecla F6.

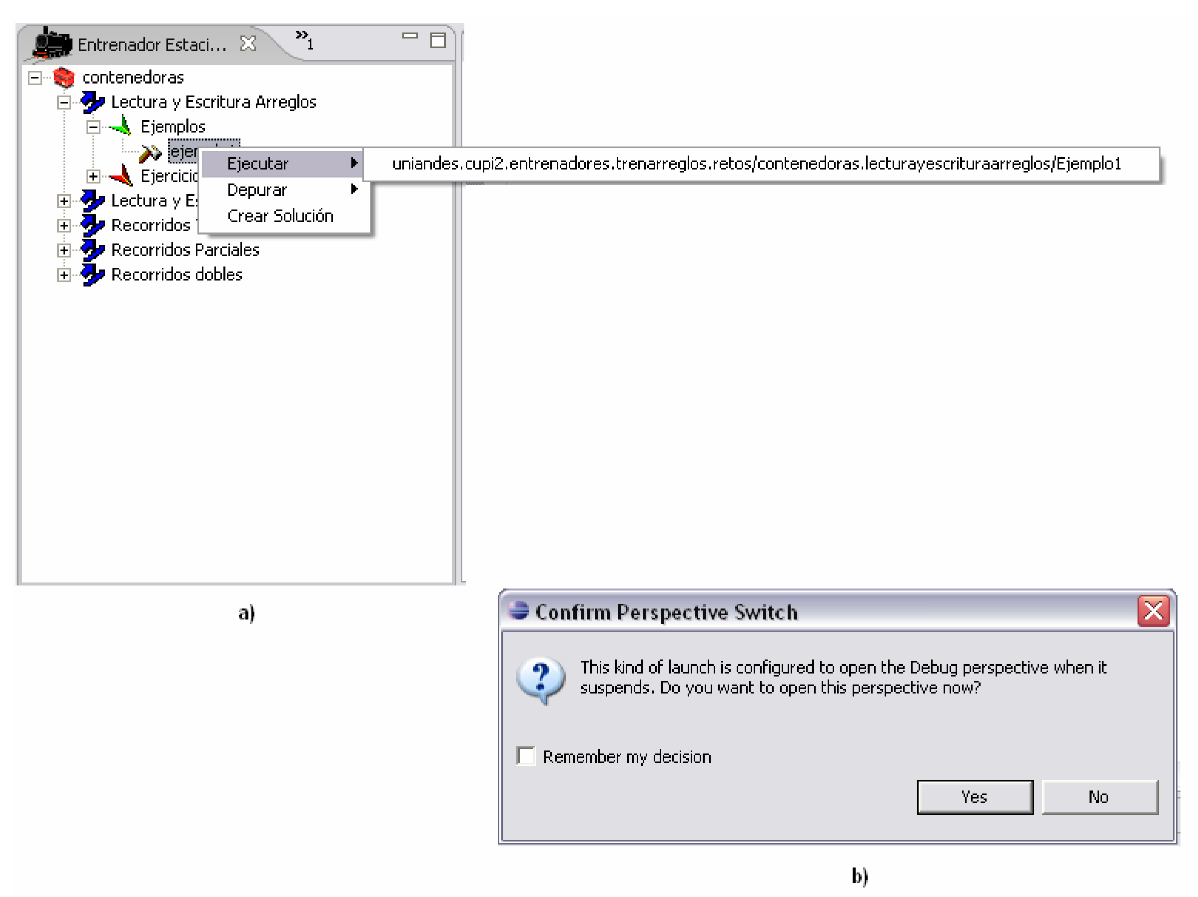


Figura 24. Componentes involucrados en el Debug de una solución: a) Explorador del entrenador. b) Dialogo para cambio de perspectiva.

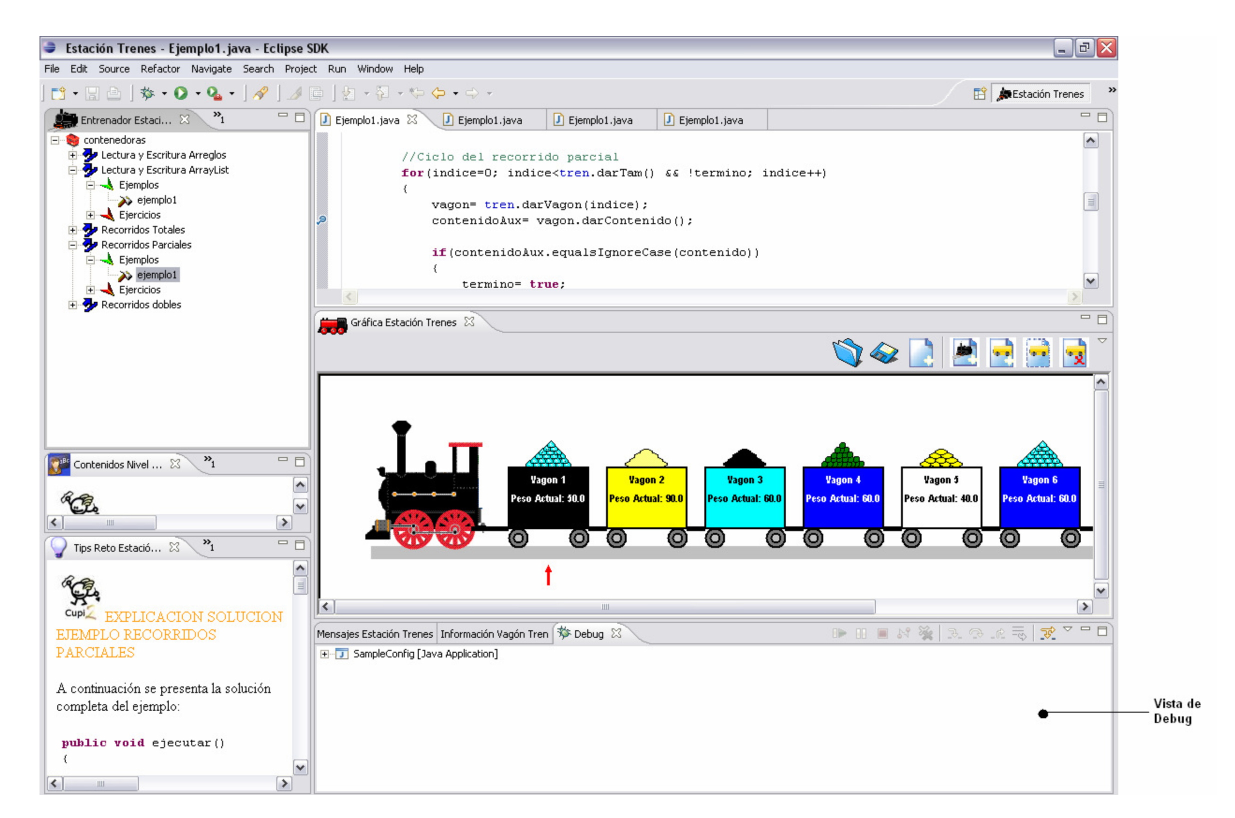


Figura 25. Perspectiva Estación Tren con la vista de Debug.

4.4 resolviendo problemas con la ejecución de soluciones

Cuando no pueda ejecutar las soluciones del entrenador, esto puede deberse a problemas con el puerto en las que éstas están siendo lanzadas. Por ésta razón es necesario cambiar el puerto en el que las soluciones son ejecutadas, por medio de las preferencias del proyecto (ver sección 6 de este manual).

5. CAMBIANDO LAS PREFERENCIAS DEL PROYECTO

El proyecto maneja dos preferencias: una para el número del puerto en el que se ejecutan las soluciones y otra para el nombre del archivo en el que se crean las soluciones por defecto de los ejemplos. El valor del puerto es un entero entre 0 y 49000; este valor solo debería cambiarse en caso de que la aplicación indique que el puerto en que se trata de correr las soluciones se encuentra ocupado.

El cambio de nombre del proyecto en el que se crean las soluciones por defecto hace que se cree un proyecto con el nuevo nombre. Las soluciones de los ejemplos se van creando en el nuevo proyecto a medida que estos van siendo seleccionados en el explorador del entrenador. El cambio del valor de esta preferencia no ocasiona que se borre ni renombre el proyecto que se había creado con el nombre anterior.

Para modificar alguna de las preferencias del proyecto siga los siguientes pasos:

1. Elija `Windows>Preferences`. Le debe aparecer una ventana similar a la siguiente:

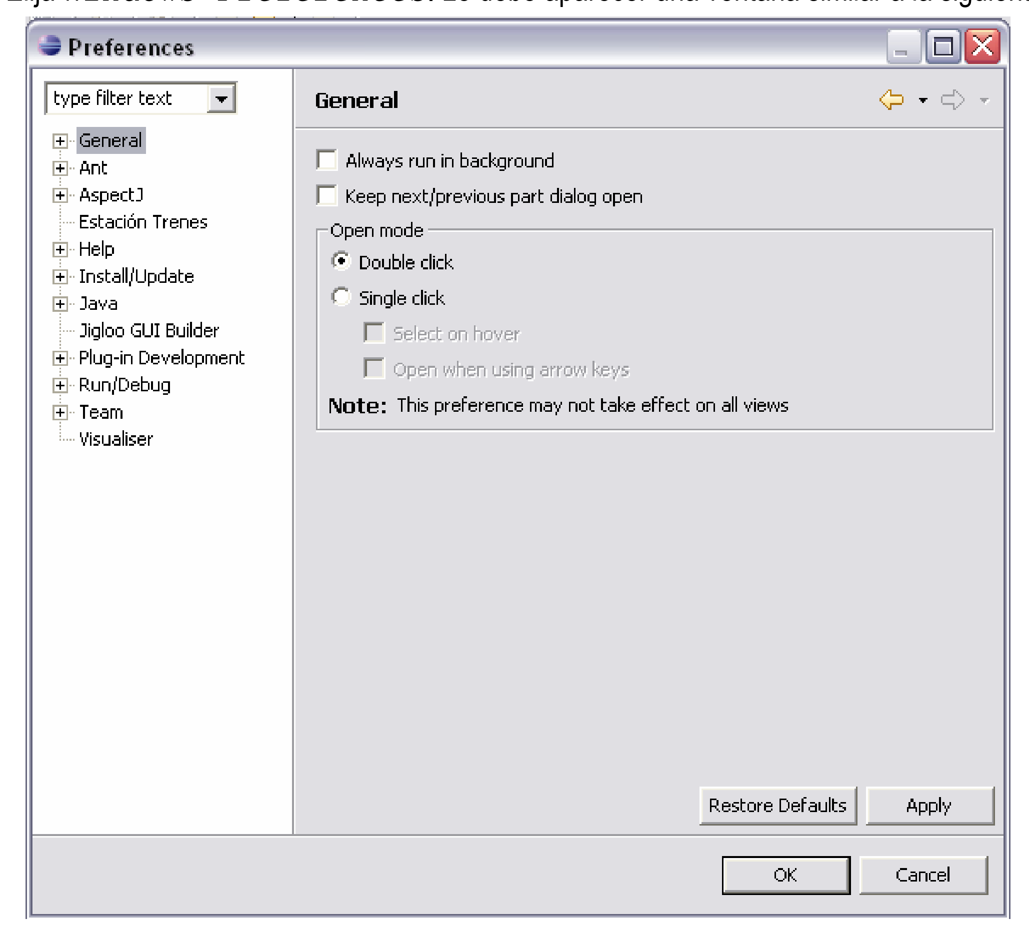


Figura 26. Ventana inicial preferencias

2. Sitúese en "Estación Trenes". Le debe aparecer una ventana similar a la que se muestra en la figura 27.

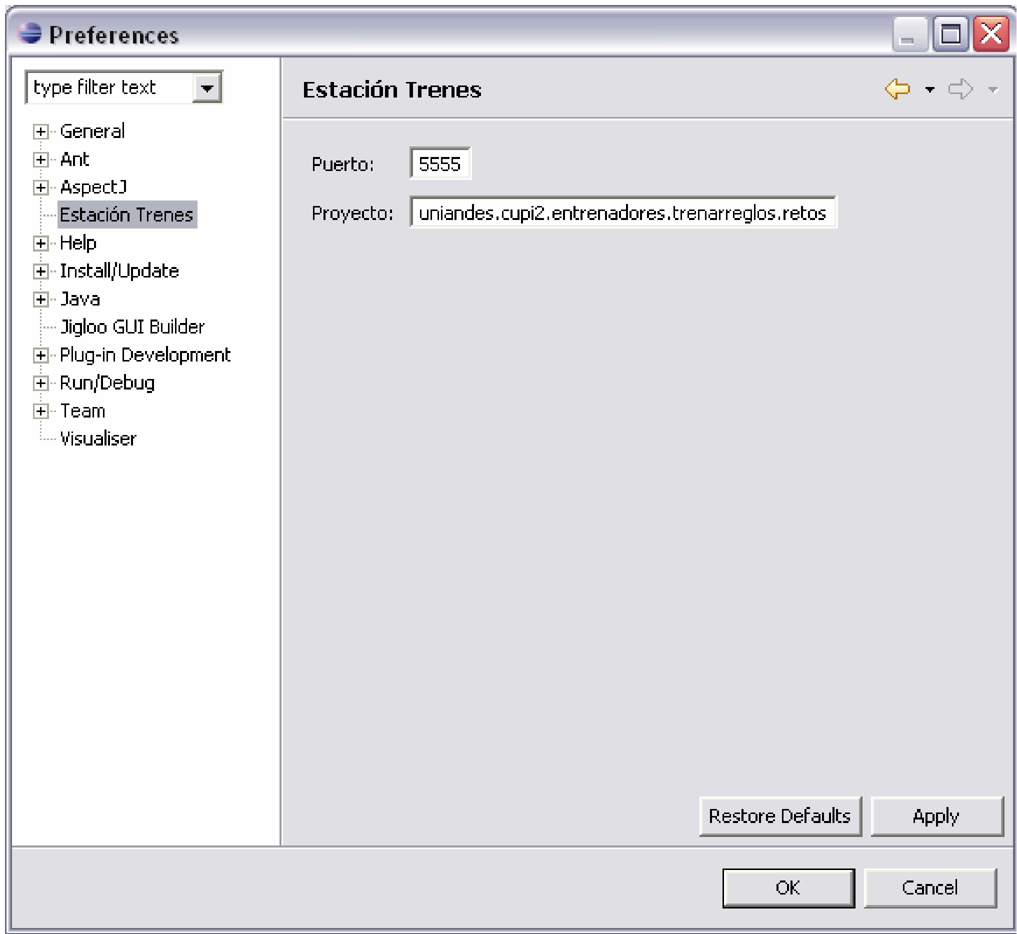


Figura 27. Ventana de las preferencias del plugin

3. Modifique el valor de una o ambas preferencias. Haga click en "Apply" y "OK" o solo en "OK".
4. El valor de las preferencias fue modificado.

Si desea restaurar los valores por defecto de ambas preferencias, haga click en "Restore Defaults", y luego haga click en "Apply".

6. API DE LA CLASE TREN Y VAGÓN

uniandes.cupi2.trenArreglos.kernel

Class Vagon

`java.lang.Object`

└ **uniandes.cupi2.trenArreglos.kernel.Vagon**

All Implemented Interfaces:
java.io.Serializable

```
public class Vagon
extends java.lang.Object
implements java.io.Serializable
```

Representa un vagón del tren

Author:
Daniel Francisco Romero

See Also:
Serialized Form

Field Summary

static int	**AMARILLO** Color amarillo
static java.lang.String	**ARENA** Contenido Arena
static int	**AZUL** Color Azul
static int	**BLANCO** Color Blanco
static int	**CYAN** Color cyan
static java.lang.String	**DIAMANTES** Contenido Diamantes
static java.lang.String	**ESMERALDAS** Contenido Esmeraldas
static int	**MAGENTA** Color magenta
static int	**NEGRO** Color Negro
static java.lang.String	**ORO** Contenido Oro
static int	**ROJO** Color Rojo
static java.lang.String	**TIERRA** Contenido Tierra
static java.lang.String	**VACIO** Contenido Vacío

static int	**VERDE** Color verde

Constructor Summary

Vagon() Constructor por defecto
Vagon(java.io.BufferedReader br) Crea un vagón a partir de los datos definidos en un archivo
Vagon(int color, java.lang.String contenido, double pesoActual, double pesoMaximo, java.lang.String nombre, double altura, double ancho) Crea un vagón con los parámetros especificados.

Method Summary

void	**cambiarAltura**(double altura) Modifica la altura del vagón
void	**cambiarAncho**(double ancho) Modifica el ancho del vagón
void	**cambiarColor**(int color) Modifica el color del vagón
void	**cambiarContenido**(java.lang.String contenido) Modifica el contenido del vagón
void	**cambiarNombre**(java.lang.String nombre) Modifica el nombre del vagón
void	**cambiarPesoActual**(double pesoActual) Modifica el peso actual del vagón
void	**cambiarPesoMaximo**(double pesoMaximo) Modifica el peso máximo del vagón
double	**darAltura**() Retorna la altura del vagón
double	**darAncho**() Retorna el ancho del vagón
int	**darColor**() Retorna el color del vagón
static int	**darColorInt**(java.lang.String color) Retorna el color especificado como un int
java.lang.String	**darColorString**() Retorna el color del vagón como un String
static java.lang.String	**darColorString**(int color) Retorna el color correspondiente al valor especificado

	como un String
java.lang.String	**darContenido**() Retorna el contenido del vagón
java.lang.String	**darNombre**() Retorna el nombre del vagón
double	**darPesoActual**() Retorna el peso actual del vagón
double	**darPesoMaximo**() Retorna el peso máximo del vagón
void	**guardar**(java.io.PrintWriter pr) Escribe el la información del vagón en un archivo

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ARENA

`public static java.lang.String` **ARENA**

Contenido Arena

TIERRA

`public static java.lang.String` **TIERRA**

Contenido Tierra

ORO

`public static java.lang.String` **ORO**

Contenido Oro

ESMERALDAS

`public static java.lang.String` **ESMERALDAS**

Contenido Esmeraldas

DIAMANTES

`public static java.lang.String` **DIAMANTES**

Contenido Diamantes

VACIO

`public static java.lang.String` **VACIO**

Contenido Vacío

ROJO

`public static int` **ROJO**

Color Rojo

BLANCO

`public static int BLANCO`

Color Blanco

NEGRO

`public static int NEGRO`

Color Negro

AZUL

`public static int AZUL`

Color Azul

CYAN

`public static int CYAN`

Color cyan

AMARILLO

`public static int AMARILLO`

Color amarillo

VERDE

`public static int VERDE`

Color verde

MAGENTA

`public static int MAGENTA`

Color magenta

Constructor Detail

Vagon

`public Vagon()`

Constructor por defecto

Vagon

```
public Vagon(int color,
             java.lang.String contenido,
             double pesoActual,
             double pesoMaximo,
             java.lang.String nombre,
             double altura,
             double ancho)
```

Crea un vagón con los parámetros especificados. En el caso de que el color o el contenido no correspondan a uno de los valores predefinidos, se coloca el valor por defecto

Parameters:

`color` - Color del vagón

`contenido` - Contenido que va a transportar el vagón
`pesoActual` - Peso del contenido que transporta el vagón
`pesoMaximo` - Peso máximo que puede transportar el vagón
`nombre` - Nombre del vagón
`altura` - Altura del vagón
`ancho` - Ancho del vagón

Vagon

`public Vagon(java.io.BufferedReader br)`
`throws EstacionException`

Crea un vagón a partir de los datos definidos en un archivo

Parameters:
`br` - Stream para leer del archivo

Throws:
`EstacionException` - Se arroja si el formato del archivo no es el esperado

Method Detail

guardar

`public void guardar(java.io.PrintWriter pr)`

Escribe el la información del vagón en un archivo

Parameters:
`pr` - Stream para escribir en el archivo

---

darAltura

`public double darAltura()`

Retorna la altura del vagón

Returns:
Altura del vagón

---

cambiarAltura

`public void cambiarAltura(double altura)`

Modifica la altura del vagón

Parameters:
`altura` - Nueva altura del vagón

---

darAncho

`public double darAncho()`

Retorna el ancho del vagón

Returns:
Ancho del vagón

---

cambiarAncho

`public void cambiarAncho(double ancho)`

Modifica el ancho del vagón

Parameters:
`ancho` - Nuevo ancho del vagón

---

darColor

```
public int darColor()
```

Retorna el color del vagón

Returns:

Color del vagón

---

cambiarColor

```
public void cambiarColor(int color)
```

Modifica el color del vagón

Parameters:

`color` - Nuevo color del vagón

---

darContenido

```
public java.lang.String darContenido()
```

Retorna el contenido del vagón

Returns:

Contenido del vagón

---

cambiarContenido

```
public void cambiarContenido(java.lang.String contenido)
```

Modifica el contenido del vagón

Parameters:

`contenido` - Nuevo contenido del vagón

---

darNombre

```
public java.lang.String darNombre()
```

Retorna el nombre del vagón

Returns:

Nombre del vagón

---

cambiarNombre

```
public void cambiarNombre(java.lang.String nombre)
```

Modifica el nombre del vagón

Parameters:

`nombre` - Nuevo nombre del vagón

---

darPesoActual

```
public double darPesoActual()
```

Retorna el peso actual del vagón

Returns:

Peso actual del vagón

---

cambiarPesoActual

```
public void cambiarPesoActual(double pesoActual)
```

Modifica el peso actual del vagón

Parameters:

`pesoActual` - Nuevo peso actual del vagón

---

darPesoMaximo
`public double` **`darPesoMaximo`**`()`

Retorna el peso máximo del vagón
Returns:
Peso máximo del vagón

---

cambiarPesoMaximo
`public void` **`cambiarPesoMaximo`**`(double pesoMaximo)`

Modifica el peso máximo del vagón
Parameters:
`pesoMaximo` - Nuevo peso máximo del vagón

---

darColorString
`public java.lang.String` **`darColorString`**`()`

Retorna el color del vagón como un String
Returns:
Color del vagón como string

---

darColorString
`public static java.lang.String` **`darColorString`**`(int color)`

Retorna el color correspondiente al valor especificado como un String
Returns:
Color como string. Si el color no existe se retorna la cadena vacía

---

darColorInt
`public static int` **`darColorInt`**`(java.lang.String color)`

Retorna el color especificado como un int
Returns:
Color como un int. El valor que se retorna por defecto es 0

uniandes.cupi2.trenArreglos.kernelCliente

Class Tren

```
java.lang.Object
  └uniandes.cupi2.trenArreglos.kernelCliente.Tren
```

All Implemented Interfaces:
java.io.Serializable
Direct Known Subclasses:
TrenAuxiliar

```
public class Tren
extends java.lang.Object
implements java.io.Serializable
```

Tren a ser manipulado por el estudiante

Author:
Daniel Francisco Romero

See Also:
Serialized Form

Constructor Summary

	Tren() Constructor sin parámetros
protected	**Tren**(int id) Construye un tren con el id especificado

Method Summary

void	**adicionarVagon**(Vagon vagon) Adiciona un vagón al tren en la posición especificada
void	**agregarMensaje**(java.lang.String mensaje) Adiciona un mensaje a la estación
void	**cambiarVagon**(Vagon vagon, int pos) Cambia el vagón del tren en la posición especificada
int	**darTam**() Retorna el tamaño del tren
Vagon	**darVagon**(int pos) Retorna el vagón especificado
void	**removerVagon**(int pos) Remueve el vagón especificado
void	**removerVagones**() Remueve todos los vagones del tren

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Tren

`public Tren()`

Constructor sin parámetros

Tren

`protected Tren(int id)`

Construye un tren con el id especificado

Parameters:

`id` - Id del tren en el lado del servidor

Method Detail

agregarMensaje

`public void agregarMensaje(java.lang.String mensaje)`

Adiciona un mensaje a la estación

Parameters:

`mensaje` - Mensaje a ser adicionado

---

adicionarVagon

`public void adicionarVagon(Vagon vagon)`

Adiciona un vagón al tren en la posición especificada

Parameters:

`vagon` - Vagón a ser adiconado

---

cambiarVagon

```
public void cambiarVagon(Vagon vagon,
                         int pos)
```

Cambia el vagón del tren en la posición especificada

Parameters:

`vagon` - Vagón por el que se va a realizar el cambio

`pos` - Posición del vagón a ser reemplazado

---

removerVagon

`public void removerVagon(int pos)`

Remueve el vagón especificado

Parameters:

`pos` - Posición del vagón a ser eliminado

---

removerVagones

`public void removerVagones()`

Remueve todos los vagones del tren

---

darVagon

`public Vagon darVagon(int pos)`

Retorna el vagón especificado

Parameters:

`posicion` - Posición del vagón a ser consultado

Returns:

Vagón a ser consultado

---

darTam

`public int darTam()`

Retorna el tamaño del tren

Returns:

Tamaño del tren

7. CRÉDITOS

Autores Plugin: Daniel Francisco Romero Acero

Asesores de Tesis: Katalina Marcos

Autor manual: Daniel Francisco Romero Acero

8. REFERENCIAS

1. Romero D., Restrepo C. (2005). Manual del Usuario- Plugin Robot Móvil.